



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION NO.                      | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|--------------------------------------|-------------|----------------------|---------------------|------------------|
| 10/714,465                           | 11/14/2003  | Jinquan Dai          | 42P17829            | 2488             |
| 45209                                | 7590        | 01/14/2009           | EXAMINER            |                  |
| INTEL/BSTZ                           |             |                      | WANG, BEN C         |                  |
| BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP |             |                      | ART UNIT            |                  |
| 1279 OAKMEAD PARKWAY                 |             |                      | PAPER NUMBER        |                  |
| SUNNYVALE, CA 94085-4040             |             |                      | 2192                |                  |
|                                      |             |                      | MAIL DATE           | DELIVERY MODE    |
|                                      |             |                      | 01/14/2009          | PAPER            |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

### Office Action Summary

**Application No.**

10/714,465

**Applicant(s)**

DAI ET AL.

**Examiner**

BEN C. WANG

**Art Unit**

2192

**Period for Reply** -- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 29 October 2008.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-36 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-4, 8-14, 18-21, 26 and 31-36 is/are rejected.
- 7) ☐ Claim(s) 5-7, 15-17, 22-25 and 27-30 is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO/SB/08)  
Paper No(s)/Mail Date \_\_\_\_\_
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: \_\_\_\_\_

**DETAILED ACTION**

1. Applicant's amendment dated October 29, 2008, responding to the Office Action mailed August 6, 2008 provided in the rejection of claims 1-36.

Claims 1-36 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Brokenshire et al.*, art made of record, as applied hereto.

***Claim Rejections – 35 USC § 103(a)***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

2. Claims 1, 11, 21, 26, 31, and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lakshmanamurthy et al. (*"Network Processor Performance Analysis Methodology"*, Aug. 15, 2002, *Intel Technology Journal*) (hereinafter 'Lakshmanamurthy') in view of Brokenshire et al. (Pat. No. US 7,392,511 B2) (hereinafter 'Brokenshire' - art made of record)

3. **As to claim 1** (Previously Presented), Lakshmanamurthy discloses a method comprising:

- configuring one or more processors into a D-stage processor pipeline (e.g., Sec. of "ABSTRACT", 1<sup>st</sup> Para – this paper describes the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP 2400 network processor, the second-generation IXA network processor; P. 19, L-Col., 3<sup>rd</sup> Para, Lines 4-9 – this methodology involves diving the application into pipeline blocks.. and latency budget for each pipeline element, and mapping the application blocks to software paradigms and the hardware resources)

Further, Lakshmanamurthy discloses the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP2400 network processor (e.g., Abstract, 1<sup>st</sup> Para), but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamically Partitioning Processing across Plurality of Heterogeneous Processors*, Brokenshire discloses the followings:

- transforming a sequential network application program into D-pipeline stages (e.g., Col. 2, Lines 23-25 – A system and method are provided to partition a computational problem based upon available processing resources in a heterogeneous processing environment and suitability to task ...; Col. 2, Lines 36-52 - ... compiles a program into at least two object files – one object file for each of the supported processor environment. During compilation, code

characteristics such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file ...) that collectively perform the an infinite packet processing state (PPS) loop of the sequential network application programs (e.g., Col. 6, Lines 15-28 - ... In operation, PU 203 schedules and orchestrates the processing of data and applications by the SPUs ...; Figs 39, 40A and 40B; Col. 17, Lines 5-23 – The ability of SPUs (Synergistic Processing Unit) to perform tasks independently under the direction of a PU (Processing Unit) enables a PU to dedicate a group of SPUs, and the memory resources associated with a group of SPUs, to performing extended tasks ... the PU can establish a dedicated pipeline relationship among a group of SPUs and their associated memory sandboxes for processing such data; Col. 19, Lines 43-54 – In lieu of an absolute timer to establish coordination among the SPUs, the PU, or one or more designated SPUs, can analyze the particular instructions or microcode being executed by an SPU in processing an spulet for problem in the coordination of the SPUs' parallel processing created by enhanced or different operating speeds ...); and

- executing the D-pipeline stages in parallel within the D-stage processor pipeline to provide the infinite parallel execution of PPS loop of the sequential network application program (e.g., Col. 2, Lines 36-52 - ... During run time, the code characteristics are combined with runtime considerations, such as the current load on the processors and the size of the data being processed, to arrive at an overall value. The overall value is then used to determine which of the

processors will be assigned the task. The values are assigned based on the characteristics of the various processors ...; Figs. 1, 39, 40B; Col. 18, Lines 10-22 – ... for processing streaming MPEG data by this dedicated pipeline. In step 4030, SPU 3908, which processes the network spulet, receives in its local storage TCP/IP data packets from network 104. In step 4032, SPU 3908 processes these TCP/IP data packets and assembles the data within these packets into software cells 102 ...; Col. 18, Lines 23-44)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Brokenshire into the Lakshmanamurthy's system to further provide other limitations stated above in the Lakshmanamurthy system.

The motivation is that it would further enhance the Lakshmanamurthy's system by taking, advancing and/or incorporating Brokenshire's system which offers significant advantages that during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations as once suggested by Brokenshire (e.g., Abstract)

4. **As to claim 11** (Previously Presented), Lakshmanamurthy discloses an article of manufacture including a machine readable medium having stored thereon instructions which may be used to program a system to perform a method, comprising:

- configuring one or more processors into a D-stage processor pipeline (e.g., Sec. of "ABSTRACT", 1<sup>st</sup> Para – this paper describes the performance analysis

methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP 2400 network processor, the second-generation IXA network processor; P. 19, L-Col., 3<sup>rd</sup> Para, Lines 4-9 – this methodology involves dividing the application into pipeline blocks, and latency budget for each pipeline element, and mapping the application blocks to software paradigms and the hardware resources)

Further, Lakshmanamurthy discloses the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP2400 network processor (e.g., Abstract, 1<sup>st</sup> Para), but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamically Partitioning Processing across Plurality of Heterogeneous Processors*, Brokenshire discloses the followings:

- transforming a sequential network application program into D-pipeline stages (e.g., Col. 2, Lines 23-25 – A system and method are provided to partition a computational problem based upon available processing resources in a heterogeneous processing environment and suitability to task ...; Col. 2, Lines 36-52 - ... compiles a program into at least two object files – one object file for each of the supported processor environment. During compilation, code characteristics such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file ...) that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application program (e.g., Col. 6, Lines 15-28 - ... In operation, PU 203

schedules and orchestrates the processing of data and applications by the SPUs ...; Figs 39, 40A and 40B; Col. 17, Lines 5-23 – The ability of SPUs (Synergistic Processing Unit) to perform tasks independently under the direction of a PU (Processing Unit) enables a PU to dedicate a group of SPUs, and the memory resources associated with a group of SPUs, to performing extended tasks ... the PU can establish a dedicated pipeline relationship among a group of SPUs and their associated memory sandboxes for processing such data; Col. 19, Lines 43-54 – In lieu of an absolute timer to establish coordination among the SPUs, the PU, or one or more designated SPUs, can analyze the particular instructions or microcode being executed by an SPU in processing an spulet for problem in the coordination of the SPUs' parallel processing created by enhanced or different operating speeds ...); and

- executing the D-pipeline stages in parallel within the D-stage processor pipeline to provide parallel execution of the infinite PPS loop of the sequential network application program (e.g., Col. 2, Lines 36-52 - ... During run time, the code characteristics are combined with runtime considerations, such as the current load on the processors and the size of the data being processed, to arrive at an overall value. The overall value is then used to determine which of the processors will be assigned the task. The values are assigned based on the characteristics of the various processors ...; Figs. 1, 39, 40B; Col. 18, Lines 10-22 – ... for processing streaming MPEG data by this dedicated pipeline. In step 4030, SPU 3908, which processes the network spulet, receives in its local



storage TCP/IP data packets from network 104. In step 4032, SPU 3908 processes these TCP/IP data packets and assembles the data within these packets into software cells 102 ...; Col. 18, Lines 23-44)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Brokenshire into the Lakshmanamurthy's system to further provide other limitations stated above in the Lakshmanamurthy system.

The motivation is that it would further enhance the Lakshmanamurthy's system by taking, advancing and/or incorporating Brokenshire's system which offers significant advantages that during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations as once suggested by Brokenshire (e.g., Abstract)

5. **As to claim 21** (Previously Presented), Lakshmanamurthy discloses a method comprising:

- constructing a flow network model from a sequential network application program (e.g., Sec. of "ABSTRACT", 1<sup>st</sup> Para – this paper describes the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP 2400 network processor, the second-generation IXA network processor; Sec. Introduction, 2<sup>nd</sup> Para, Lines 18-23 - .. in analyzing the performance of networking applications

running on the IXP2400 network processor and presents a case study using the IPv4 forwarding + DiffServ application; 3<sup>rd</sup> Para - ... a detailed data movement model of the target application. This model describes the various operations performed by the network processor on every received packet);

- cutting the flow network model into a plurality of preliminary pipeline stages (e.g., P. 19, L-Col., 3<sup>rd</sup> Para, Lines 4-9 – this methodology involves diving the application into pipeline blocks.. and latency budget for each pipeline element, and mapping the application blocks to software paradigms and the hardware resources)

Further, Lakshmanamurthy discloses the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP2400 network processor (e.g., Abstract, 1<sup>st</sup> Para) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamically Partitioning Processing across Plurality of Heterogeneous Processors*, Brokenshire discloses:

- transforming the preliminary pipeline stages to perform control flow and variable transmission therebetween to form D-pipeline stages (e.g., Col. 2, Lines 23-25 – A system and method are provided to partition a computational problem based upon available processing resources in a heterogeneous processing environment and suitability to task ...; Col. 2, Lines 36-52 - ... compiles a program into at least two object files – one object file for each of the supported processor environment. During compilation, code

characteristics such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file ...) that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application program to enable parallel execution of the infinite PPS loop of the sequential network application program (e.g., Col. 6, Lines 15-28 - ... In operation, PU 203 schedules and orchestrates the processing of data and applications by the SPUs ...; Figs 39, 40A and 40B; Col. 17, Lines 5-23 – The ability of SPUs (Synergistic Processing Unit) to perform tasks independently under the direction of a PU (Processing Unit) enables a PU to dedicate a group of SPUs, and the memory resources associated with a group of SPUs, to performing extended tasks ... the PU can establish a dedicated pipeline relationship among a group of SPUs and their associated memory sandboxes for processing such data; Col. 19, Lines 43-54 – In lieu of an absolute timer to establish coordination among the SPUs, the PU, or one or more designated SPUs, can analyze the particular instructions or microcode being executed by an SPU in processing an spulet for problem in the coordination of the SPUs' parallel processing created by enhanced or different operating speeds ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Brokenshire into the Lakshmanamurthy's system to further provide other limitations stated above in the Lakshmanamurthy system.

The motivation is that it would further enhance the Lakshmanamurthy's system by taking, advancing and/or incorporating Brokenshire's system which offers significant advantages that during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations as once suggested by Brokenshire (e.g., Abstract)

6. **As to claim 26** (Currently Amended), Lakshmanamurthy discloses an article of manufacture including a machine readable medium having stored thereon instructions which may be used to program a system to perform a method, comprising:

- constructing a flow network model from a sequential network application program (e.g., Sec. of "ABSTRACT", 1<sup>st</sup> Para – this paper describes the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP 2400 network processor; the second-generation IXA network processor; Sec. Introduction, 2<sup>nd</sup> Para, Lines 18-23 - .. in analyzing the performance of networking applications running on the IXP2400 network processor and presents a case study using the IPv4 forwarding + DiffServ application; 3<sup>rd</sup> Para - ... a detailed data movement model of the target application. This model describes the various operations performed by the network processor on every received packet);
- cutting the flow network model into a plurality of preliminary pipeline stages (e.g., P. 19, L-Col., 3<sup>rd</sup> Para, Lines 4-9 – this methodology involves diving the

application into pipeline blocks.. and latency budget for each pipeline element, and mapping the application blocks to software paradigms and the hardware resources)

Further, Lakshmanamurthy discloses the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP2400 network processor (e.g., Abstract, 1<sup>st</sup> Para) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamically Partitioning Processing across Plurality of Heterogeneous Processors*, Brokenshire discloses:

- transforming the preliminary pipeline stages to perform control flow and variable transmission therebetween in order to form D-pipeline stages (e.g., Col. 2, Lines 23-25 – A system and method are provided to partition a computational problem based upon available processing resources in a heterogeneous processing environment and suitability to task ...; Col. 2, Lines 36-52 - ... compiles a program into at least two object files – one object file for each of the supported processor environment. During compilation, code characteristics such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file ...) that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application program to enable parallel execution of the infinite PPS loop of the sequential network application program (e.g., Col. 6, Lines 15-28 - ... In operation, PU 203 schedules and orchestrates the processing of data

and applications by the SPUs ...; Figs 39, 40A and 40B; Col. 17, Lines 5-23 – The ability of SPUs (Synergistic Processing Unit) to perform tasks independently under the direction of a PU (Processing Unit) enables a PU to dedicate a group of SPUs, and the memory resources associated with a group of SPUs, to performing extended tasks ... the PU can establish a dedicated pipeline relationship among a group of SPUs and their associated memory sandboxes for processing such data; Col. 19, Lines 43-54 – In lieu of an absolute timer to establish coordination among the SPUs, the PU, or one or more designated SPUs, can analyze the particular instructions or microcode being executed by an SPU in processing an spulet for problem in the coordination of the SPUs' parallel processing created by enhanced or different operating speeds ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Brokenshire into the Lakshmanamurthy's system to further provide other limitations stated above in the Lakshmanamurthy system.

The motivation is that it would further enhance the Lakshmanamurthy's system by taking, advancing and/or incorporating Brokenshire's system which offers significant advantages that during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations as once suggested by Brokenshire (e.g., Abstract)

7. **As to claim 31** (Currently Amended), Lakshmanamurthy discloses an apparatus, comprising:

- a processor (e.g., Sec. of "ABSTRACT", 1<sup>st</sup> Para – this paper describes the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP 2400 network processor, the second-generation IXA network processor; Fig. 1 – IXP 2400 external interface, element of "IXP 2400"; P. 20, R-Col., 1<sup>st</sup> Para; P. 21, L-Col., 3<sup>rd</sup> Para – IXP 2400 contains eight multi-threaded, packet-processing micro-engines; these micro-engines are highly programmable packet processors and support multi threading of up to eight threads each; each micro-engine provides a variety of network processing functions in hardware; P. 21, R-Col., 2<sup>nd</sup> Para – the IXP 2400 also has an integrated low-power general-purpose Intel® Xscale™ micro-architecture core; the integrated Xscale™ process offers ample processing power for running control plane software);
- a memory coupled to the processor (e.g., P. 20, L-Col., 4<sup>th</sup> Para – extern DRAM and SRAM; P. 20, L-Col., 4<sup>th</sup> Para – P. 21, L-Col., 1<sup>st</sup> Para – the SRAM is primarily used for packet descriptors, queue descriptors, counters, and other data structures; Fig. 2 – IXP 2400 internal architecture, elements of "QDR SRAM", "DDRAM"; Fig. 3 – IXP 2400-based OC-48 line card configuration, element of "DDR SDRAM")

Further, Lakshmanamurthy discloses the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP2400 network processor (e.g., Abstract, 1<sup>st</sup> Para) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamically Partitioning Processing across Plurality of Heterogeneous Processors*, Brokenshire discloses:

- the memory including a compiler to cause transformation of a sequential network application program into D-pipeline stages (e.g., Col. 2, Lines 23-25 – A system and method are provided to to partition a computational problem based upon available processing resources in a heterogeneous processing environment and suitability to task ...; Col. 2, Lines 36-52 - ... compiles a program into at least two object files – one object file for each of the supported processor environment. During compilation, code characteristics such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file ...) that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application program to enable parallel execution of the D-pipeline stages within a D-stage processor pipeline to provide parallel execution of the infinite PPS loop of the sequential network application program (e.g., Col. 6, Lines 15-28 - ... In operation, PU 203 schedules and orchestrates the processing of data and applications by the SPUs ...; Figs 39, 40A and 40B; Col. 17, Lines 5-23 – The ability of SPUs (Synergistic Processing Unit) to perform tasks independently



under the direction of a PU (Processing Unit) enables a PU to dedicate a group of SPUs, and the memory resources associated with a group of SPUs, to performing extended tasks ... the PU can establish a dedicated pipeline relationship among a group of SPUs and their associated memory sandboxes for processing such data; Col. 19, Lines 43-54 – In lieu of an absolute timer to establish coordination among the SPUs, the PU, or one or more designated SPUs, can analyze the particular instructions or microcode being executed by an SPU in processing an spulet for problem in the coordination of the SPUs' parallel processing created by enhanced or different operating speeds ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Brokenshire into the Lakshmanamurthy's system to further provide other limitations stated above in the Lakshmanamurthy system.

The motivation is that it would further enhance the Lakshmanamurthy's system by taking, advancing and/or incorporating Brokenshire's system which offers significant advantages that during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations as once suggested by Brokenshire (e.g., Abstract)

8. **As to claim 34** (Currently Amended), Lakshmanamurthy discloses a system comprising:

- a processor (e.g., Sec. of "ABSTRACT", 1<sup>st</sup> Para – this paper describes the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP 2400 network processor, the second-generation IXA network processor; Fig. 1 – IXP 2400 external interface, element of "IXP 2400"; P. 20, R-Col., 1<sup>st</sup> Para; P. 21, L-Col., 3<sup>rd</sup> Para – IXP 2400 contains eight multi-threaded, packet-processing micro-engines; these micro-engines are highly programmable packet processors and support multi threading of up to eight threads each; each micro-engine provides a variety of network processing functions in hardware; P. 21, R-Col., 2<sup>nd</sup> Para – the IXP 2400 also has an integrated low-power general-purpose Intel® Xscale™ micro-architecture core; the integrated Xscale™ process offers ample processing power for running control plane software);
- a memory controller coupled to the processor (e.g., P. 21, L-Col., 3<sup>rd</sup> Para, Lines 13-14 – the memory controllers facilitate efficient access to the of-chip SRAM and DRAM); and
- a DDR SRAM memory coupled to the processor (e.g., P. 20, L-Col., 4<sup>th</sup> Para – extern DRAM and SRAM; P. 20, L-Col., 4<sup>th</sup> Para – P. 21, L-Col., 1<sup>st</sup> Para – the SRAM is primarily used for packet descriptors, queue descriptors, counters, and other data structures; Fig. 2 – IXP 2400 internal architecture, element of "QDR SRAM"; Fig. 3 – IXP 2400-based OC-48 line card configuration, element of "DDR SDRAM")

Further, Lakshmanamurthy discloses the performance analysis methodology developed to analyze the performance of various networking applications that are targeted for running on the IXP2400 network processor (e.g., Abstract, 1<sup>st</sup> Para) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Dynamically Partitioning Processing across Plurality of Heterogeneous Processors*, Brokenshire discloses:

- the memory including a compiler to cause transformation of a sequential network application program into D-application program stages (e.g., Col. 2, Lines 23-25 – A system and method are provided to partition a computational problem based upon available processing resources in a heterogeneous processing environment and suitability to task ...; Col. 2, Lines 36-52 - ... compiles a program into at least two object files – one object file for each of the supported processor environment. During compilation, code characteristics such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file ...) that collectively perform an infinite packet processing stage (PPS) loop of the sequential network application program to enable parallel execution of the D-application program stages within a D-stage processor pipeline to provide parallel execution of the infinite PPS loop of the sequential network application program (e.g., Col. 6, Lines 15-28 - ... In operation, PU 203 schedules and orchestrates the processing of data and applications by the SPUs ...; Figs 39, 40A and 40B; Col. 17, Lines 5-23 – The ability of SPUs (Synergistic

Processing Unit) to perform tasks independently under the direction of a PU (Processing Unit) enables a PU to dedicate a group of SPUs, and the memory resources associated with a group of SPUs, to performing extended tasks ... the PU can establish a dedicated pipeline relationship among a group of SPUs and their associated memory sandboxes for processing such data; Col. 19, Lines 43-54 – In lieu of an absolute timer to establish coordination among the SPUs, the PU, or one or more designated SPUs, can analyze the particular instructions or microcode being executed by an SPU in processing an spulet for problem in the coordination of the SPUs' parallel processing created by enhanced or different operating speeds ...)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Brokenshire into the Lakshmanamurthy's system to further provide other limitations stated above in the Lakshmanamurthy system.

The motivation is that it would further enhance the Lakshmanamurthy's system by taking, advancing and/or incorporating Brokenshire's system which offers significant advantages that during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations as once suggested by Brokenshire (e.g., Abstract)

9. Claims 2, 12, 32, and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lakshmanamurthy in view of Brokenshire and Rakhmatov et al., ("*Hardware-Software Bipartitioning for Dynamically Reconfigurable Systems*", May 2002, *ACM*) (hereinafter 'Rakhmatov')

10. **As to claim 2** (Original) (incorporating the rejection in claim 1), Lakshmanamurthy discloses network processor performance analysis methodology (e.g., Sec. of "ABSTRACT", 3<sup>rd</sup> Para) and Brokenshire discloses during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations (e.g., Abstract), but Lakshmanamurthy and Brokenshire do not explicitly disclose other limitations stated below.

However, in an analogous art of *hardware-software bi-partitioning for dynamically reconfigurable systems*, Rakhmatov discloses transforming the sequential application program comprises constructing a flow network model for the sequential application program; selecting a plurality of preliminary pipeline stages from the flow network model; and modifying the preliminary pipeline stages to perform control flow and variable transmission therebetween to form the D-pipeline stages (e.g., Sec. of "ABSTRACT" – a method for mapping nodes of an application control flow graph either to software or reconfigurable hardware, explicitly targeting minimization of the energy-delay cost due to both computation and configuration; using network flow techniques,

after transforming the original control flow graph into an equivalent network; P. 145, R-Col., 1<sup>st</sup> Para through 3<sup>rd</sup> Para – the software can directly configure the hardware, which is partially reconfigurable; partial reconfiguration allows for a selective change of hardware segments of arbitrary size at an arbitrary location, without disrupting the operation of the rest of the hardware space; such a capability greatly reduces reconfiguration time and energy consumption, because the hardware updates are highly localized. Three types of problems: (1) energy-delay product minimization, (2) energy minimization under the delay constraint, and (3) delay minimization under the energy constraint can be resolved via using network flow techniques; specifically, the cost of a node depends whether it is in software or in hardware and the cost of an edge depends whether its origin node is in software or in hardware and whether its destination node is in software or in hardware; P. 146, L-Col., 2<sup>nd</sup> Para – 4<sup>th</sup> Para – the cost/weight can be either the energy or the delay or the energy-delay product of a node/edge, weighted by its execution frequency; first, transferring control from a hardware block to a software block is more expensive than transferring control from a software block to a software block; second, transferring control from a software block to a hardware block is more expensive than transferring control from a hardware block to a hardware block; P. 147, Sec. of “Constrained Bipartitioning Algorithms” – cost-driven constrained bi-partitioning; weight-driven constrained bi-partitioning; P. 148, R-Col., 1<sup>st</sup> Para; Fig. 3 – proposed constrained bi-partitioning algorithms)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rakhmatov into the

Lakshmanamurthy-Brokenshire's system to further provide other limitations stated above in the Lakshmanamurthy-Brokenshire system.

The motivation is that it would further enhance the Lakshmanamurthy-Brokenshire's system by taking, advancing and/or incorporating Rakhmatov's system which offers significant advantages for providing an efficient bi-partitioning algorithm that finds an optimal solution for energy-delay product minimization and systematically searches for the best in polynomially bounded set of good solutions for delay-constrained energy minimization, and energy-constrained delay minimization, the formulation also including costs and weights as design parameters as once suggested by Rakhmatov (e.g., Sec. of "CONCLUSION")

11. **As to claim 12** (Previously Presented) (incorporating the rejection in claim 11), please refer to claim 2 as set forth accordingly.

12. **As to claim 32** (Original) (incorporating the rejection in claim 31), Lakshmanamurthy discloses network processor performance analysis methodology (e.g., Sec. of "ABSTRACT", 3<sup>rd</sup> Para) and Brokenshire discloses during compilation, code characteristics, such as data locality, computational intensity, and data parallelism, are analyzed and recorded in the object file; and, during run time, the code characteristics are combined with runtime considerations (e.g., Abstract) but Lakshmanamurthy and Brokenshire do not explicitly disclose other limitations stated below.

However, in an analogous art of *hardware-software bi-partitioning for dynamically reconfigurable systems*, Rakhmatov discloses the compiler to cause construction of a flow network model for the sequential application program, to cause selection of a plurality of preliminary pipeline stages from the flow network model and to cause modification of the preliminary pipeline stages to perform control flow and variable transformation therebetween to form the D-pipeline stages (e.g., Sec. of "ABSTRACT" – a method for mapping nodes of an application control flow graph either to software or reconfigurable hardware, explicitly targeting minimization of the energy-delay cost due to both computation and configuration; using network flow techniques, after transforming the original control flow graph into an equivalent network; P. 145, R-Col., 1<sup>st</sup> Para through 3<sup>rd</sup> Para – the software can directly configure the hardware, which is partially reconfigurable; partial reconfiguration allows for a selective change of hardware segments of arbitrary size at an arbitrary location, without disrupting the operation of the rest of the hardware space; such a capability greatly reduces reconfiguration time and energy consumption, because the hardware updates are highly localized. Three types of problems: (1) energy-delay product minimization, (2) energy minimization under the delay constraint, and (3) delay minimization under the energy constraint can be resolved via using network flow techniques; specifically, the cost of a node depends whether it is in software or in hardware and the cost of an edge depends whether its origin node is in software or in hardware and whether its destination node is in software or in hardware; P. 146, L-Col., 2<sup>nd</sup> Para – 4<sup>th</sup> Para – the cost/weight can be either the energy or the delay or the energy-delay product of a node/edge, weighted by its



execution frequency; first, transferring control from a hardware block to a software block is more expensive than transferring control from a software block to a software block; second, transferring control from a software block to a hardware block is more expensive than transferring control from a hardware block to a hardware block; P. 147, Sec. of "Constrained Bipartitioning Algorithms" – cost-driven constrained bi-partitioning; weight-driven constrained bi-partitioning; P. 148, R-Col., 1<sup>st</sup> Para; Fig. 3 – proposed constrained bi-partitioning algorithms)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Rakhmatov into the Lakshmanamurthy-Brokenshire's system to further provide other limitations stated above in the Lakshmanamurthy-Brokenshire system.

The motivation is that it would further enhance the Lakshmanamurthy-Brokenshire's system by taking, advancing and/or incorporating Rakhmatov's system which offers significant advantages for providing an efficient bi-partitioning algorithm that finds an optimal solution for energy-delay product minimization and systematically searches for the best in polynomially bounded set of good solutions for delay-constrained energy minimization, and energy-constrained delay minimization, the formulation also including costs and weights as design parameters as once suggested by Rakhmatov (e.g., Sec. of "CONCLUSION")

13. **As to claim 35** (Original) (incorporating the rejection in claim 34), please refer to claim 32 as set forth accordingly.

14. Claims 3-4, 8, 10, 13-14, 18, 20, 33, and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lakshmanamurthy in view of Brokenshire, Rakhmatov and Robschink et al., ("*Efficient Path Conditions in Dependence Graphs*", May 2002, ACM) (hereinafter 'Robschink')

15. **As to claim 3** (Original) (incorporating the rejection in claim 2), Rakhmatov discloses constructing the flow network model (e.g., Sec. of "ABSTRACT" – a method for mapping nodes of an application control flow graph either to software or reconfigurable hardware, explicitly targeting minimization of energy-delay cost due to both computation and configuration; show how these problems can be tackled by using network flow techniques, after transforming the original control flow graph into an equivalent network), but Lakshmanamurthy, Brokenshire and Rakhmatov do not explicitly disclose other limitations stated below.

However, in an analogous art of *efficient path conditions in dependence graphs*, Robschink discloses transforming the application program into a static, single-assignment form (e.g., Sec. 2.2 – Path conditions, 2<sup>nd</sup> Pa. – since there may be assignments to the same variable at different program points, all programs must be transformed into static single assignment form (SSA) first. In SSA form, there is at most one assignment to every variable. If necessary, we will distinguish different SSA-variants of a program variable by additional indices; P. 480, 3<sup>rd</sup> Para – since the program is transformed to SSA form first, some additional constraints must be

Art Unit: 2192

generated which represent the  $\Phi$ -function occurring in SSA form); building a control flow graph for a loop body of the application program; building a dependence graph based on a summary graph of the control flow graph (e.g., Sec. 1 – Introduction, 4<sup>th</sup> Para – *ValSoft* can build a dependence graph for 50000 lines of C; forward and backward slices or chops can be interactively computed and visualized in the source text; Fig. 1 – a *mergesort* program and part of its SDG (System Dependence Graph); Sec. 2.1 – Dependence and slices, 2<sup>nd</sup> Para – slices can be defined via the system dependence graph (SDG); Sec. 3 – Basic Analysis, 1<sup>st</sup> Para; Sec. 3.1 – Analyzing data flow, 1<sup>st</sup> Para) and identified, strongly-connected components (SSC) of the control flow graph; and constructing the flow network model according to a summary graph of the dependence graph and identified SSC nodes of the dependence graph (e.g., Sec. 4.2 – Exploiting interval analysis, 2<sup>nd</sup> Par through 3<sup>rd</sup> Para)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Robschink into the Lakshmanamurthy-Brokenshire-Rakhmatov's system to further provide other limitations stated above in the Lakshmanamurthy-Brokenshire-Rakhmatov system.

The motivation is that it would further enhance the Lakshmanamurthy-Brokenshire-Rakhmatov's system by taking, advancing and/or incorporating Robschink's system which offers that path conditions in dependence graphs are a valuable tool for various kinds of program analysis, such as program understanding or safety checks as once suggested by Robschink (e.g., Sec. of "CONCLUSION AND FUTURE WORK", 1<sup>st</sup> Para)

16. **As to claim 4** (Original) (incorporating the rejection in claim 3), Rakhmatov discloses constructing the flow network model comprises assigning a unique source node and a unique sink node to the flow network model (e.g., Sec. 4 – Proposed Solution, 1<sup>st</sup> Para)

Robschink discloses adding a program node to the flow network model for each SSC node identified in the summary graph of the dependence graph (e.g., Sec. 2.2 – Path conditions, 2<sup>nd</sup> Pa. – since there may be assignments to the same variable at different program points, all programs must be transformed into static single assignment form (SSA) first. In SSA form, there is at most one assignment to every variable. If necessary, we will distinguish different SSA-variants of a program variable by additional indices; P. 480, 3<sup>rd</sup> Para – since the program is transformed to SSA form first, some additional constraints must be generated which represent the  $\Phi$ -function occurring in SSA form); adding a variable node to the flow network model for each variable that is defined and used by multiple program nodes; adding a control node C to the flow network model for each SSC node identified in the summary graph of the dependence graph as a source of control dependence (e.g., Sec. 1 – Introduction, 4<sup>th</sup> Para – *ValSoft* can build a dependence graph for 50000 lines of C; forward and backward slices or chops can be interactively computed and visualized in the source text; Fig. 1 – a *mergesort* program and part of its SDG (System Dependence Graph); Sec. 2.1 – Dependence and slices, 2<sup>nd</sup> Para – slices can be defined via the system dependence graph (SDG); Sec. 3 – Basic Analysis, 1<sup>st</sup> Para; Sec. 3.1 – Analyzing data flow, 1<sup>st</sup> Para)

Further, Rakhmatov discloses generating edges having an associated weight to connect corresponding program nodes to corresponding variable nodes; generating edges having an associated weight to connect corresponding program nodes to corresponding control nodes; and generating edges between the program nodes and one of the source node and the sink node (e.g., Sec. 2 – Problem Description, 1<sup>st</sup> Para through 4<sup>th</sup> Para, and 6<sup>th</sup> Para; P. 147, L-Col., 1<sup>st</sup> Para; Sec. 4 – Proposed Solution, 1<sup>st</sup> Para through 2<sup>nd</sup> Para; P. 147, R-Col., 1<sup>st</sup> Para through 4<sup>th</sup> Para)

17. **As to claim 8** (Original) (incorporating the rejection in claim 2), Rakhmatov discloses selecting the plurality of preliminary pipeline stages comprises cutting the flow network model into D-1 successive cuts, such that each cut is a balanced minimum cost cut (e.g., Sec. 1 – Introduction, 4<sup>th</sup> Para – cost function involve costs of all nodes and all edges in the CFG, and not just the edges in the cut-set separating software-mapped nodes and hardware-mapped nodes; specifically, the cost of a node depends whether it is in software or in hardware, and the cost of an edge depends whether its origin node is in software or in hardware and whether its destination node is in software or in hardware; Sec. 3 – Related Work, 2<sup>nd</sup> Para (Circuit Partitioning) Through 3<sup>rd</sup> Para, 6<sup>th</sup> Para (Our Contribution) – our contribution is to show how a CFG (Control Flow Graph) with node and edge costs can be transformed into a network, so that a minimum cut in the network corresponds to an optimal bi-partition of the CFG; P. 147, 1<sup>st</sup> Para through 2<sup>nd</sup> Para; Fig. 2 – unconstrained bi-partitioning algorithm –  $CUT = FindMinCut(V, E)$ )

18. **As to claim 10** (Previously Presented) (incorporating the rejection in claim 2), Rakhmatov discloses modifying the preliminary pipeline stages comprises (a) selecting a preliminary pipeline stage; (b) altering the selected preliminary pipeline stage to enable proper transmission of live variables; and control flow to and from the selected preliminary pipeline stage; and (c) (a) – (b) for each preliminary pipeline stage to form the D-pipeline stages of a parallel network application (e.g., Sec. of “ABSTRACT” – a method for mapping nodes of an application control flow graph either to software or reconfigurable hardware, explicitly targeting minimization of the energy-delay cost due to both computation and configuration; using network flow techniques, after transforming the original control flow graph into an equivalent network; P. 145, R-Col., 1<sup>st</sup> Para Through 3<sup>rd</sup> Para – the software can directly configure the hardware, which is partially reconfigurable; partial reconfiguration allows for a selective change of hardware segments of arbitrary size at an arbitrary location, without disrupting the operation of the rest of the hardware space; such a capability greatly reduces reconfiguration time and energy consumption, because the hardware updates are highly localized. Three types of problems: (1) energy-delay product minimization, (2) energy minimization under the delay constraint, and (3) delay minimization under the energy constraint can be resolved via using network flow techniques; specifically, the cost of a node depends whether it is in software or in hardware and the cost of an edge depends whether its origin node is in software or in hardware and whether its destination node is in software or in hardware; P. 146, L-Col., 2<sup>nd</sup> Para – 4<sup>th</sup> Para – the cost/weight can be either the energy or the delay or the energy-delay product of a node/edge, weighted by its

execution frequency; first, transferring control from a hardware block to a software block is more expensive than transferring control from a software block to a software block; second, transferring control from a software block to a hardware block is more expensive than transferring control from a hardware block to a hardware block; P. 147, Sec. of "Constrained Bipartitioning Algorithms" – cost-driven constrained bi-partitioning; weight-driven constrained bi-partitioning; P. 148, R-Col., 1<sup>st</sup> Para; Fig. 3 – proposed constrained bi-partitioning algorithms)

19. **As to claim 13** (Original) (incorporating the rejection in claim 12), please refer to claim 3 as set forth accordingly.

20. **As to claim 14** (Original) (incorporating the rejection in claim 13), please refer to claim 4 as set forth accordingly.

21. **As to claim 18** (Original) (incorporating the rejection in claim 12), please refer to claim 8 as set forth accordingly.

22. **As to claim 20** (Original) (incorporating the rejection in claim 12), please refer to claim 10 as set forth accordingly.

23. **As to claim 33** (Original) (incorporating the rejection in claim 32), Robschink discloses the compiler to cause D-1 successive cuts of the flow network mode, such

that each cut is a balanced, minimum cost cut to form the D-preliminary pipeline stages (e.g., Sec. 1 – Introduction, 4<sup>th</sup> Para – cost function involve costs of all nodes and all edges in the CFG, and not just the edges in the cut-set separating software-mapped nodes and hardware-mapped nodes; specifically, the cost of a node depends whether it is in software or in hardware, and the cost of an edge depends whether its origin node is in software or in hardware and whether its destination node is in software or in hardware; Sec. 3 – Related Work, 2<sup>nd</sup> Para (Circuit Partitioning) Through 3<sup>rd</sup> Para, 6<sup>th</sup> Para (Our Contribution) – our contribution is to show how a CFG (Control Flow Graph) with node and edge costs can be transformed into a network, so that a minimum cut in the network corresponds to an optimal bi-partition of the CFG; P. 147, 1<sup>st</sup> Para through 2<sup>nd</sup> Para; Fig. 2 – unconstrained bi-partitioning algorithm –  $CUT = FindMinCut(V,E)$ )

24. **As to claim 36** (Original) (incorporating the rejection in claim 35), please refer to claim 33 as set forth accordingly.

25. Claims 9 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lakshmanamurthy in view of Brokenshire, Rakhmatov and Robschink and Goldberg et al., ("*A New Approach to the Maximum-Flow Problem*", 1988, *ACM*) (hereinafter 'Goldberg')



26. **As to claim 9** (Original) (incorporating the rejection in claim 8),  
Lakshmanamurthy, Brokenshire, Rakhmatov, and Robschink do not disclose other  
limitations stated below.

However, in an analogous art of *a new approach to the maximum-flow problem*,  
Goldberg discloses cutting is performed using an iterative balanced to push-relabel  
algorithm (e.g., P. 922, 4<sup>th</sup> Para – the algorithm pushes flow through the network to find  
a blocking flow, which determines the acyclic network for the next phase; our algorithm  
maintains a pre-flow in the original network and pushes local flow excess toward the  
sink along what it estimates to be shortest paths in the residual graph; P. 924, 4<sup>th</sup> Para –  
the pre-flow algorithm works by examining vertices other than *s* and *t* with positive flow  
excess and pushing excess from them to vertices estimated to be closer to the sink *t*,  
with the goal of getting as much excess as possible to *t*.)

Therefore, it would have been obvious to one of ordinary skill in the art, at the  
time the invention was made to combine the teachings of Goldberg into the  
Lakshmanamurthy-Brokenshire-Rakhmatov-Robschink's system to further provide other  
limitations stated above in the Lakshmanamurthy-Brokenshire-Rakhmatov-Robschink  
system.

The motivation is that it would further enhance the Lakshmanamurthy-  
Rakhmatov-Brokenshire-Robschink's system by taking, advancing and/or incorporating  
Goldberg's system which offers significant advantages that the method maintains a pre-  
flow in the original network and pushes local flow excess toward the sink along what are  
estimated to be shortest paths as once suggested by Goldberg (e.g., P. 921, 1<sup>st</sup> Para)

27. **As to claim 19** (Original) (incorporating the rejection in claim 18), please refer to claim 9 as set forth accordingly.

***Allowable Subject Matter***

28. Claims 5-7, 15-17, 22-25 and 27-30 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten to overcome the rejections under 35 U.S.C. 103(a) set forth in this office action and to include all the limitations of the base claim and any intervening claims.

The following is an examiner's statement of reasons for allowance:

Regarding claims 5-7, 15-17, 22-25, and 27-30, prior art of record fails to reasonably show or suggest the specific edge generations having associated weights, transformation of the preliminary application program stage, and transformation of the control flow as claimed. Specifically, the methods to generate edges having an associated weight to connect corresponding program nodes to (1) corresponding variable nodes, (2) corresponding controls nodes; the method to generate the edges between program nodes and one of the source node and the sink nodes; transformation of the preliminary application program stages; and transformation of the control flow in details.

***Conclusion***

29. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-

1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/  
Ben C. Wang  
Examiner, Art Unit 2192

/Tuan Q. Dam/  
Supervisory Patent Examiner, Art Unit 2192